Лабораторна робота 5

РОЗРОБКА ПРОСТИХ НЕЙРОННИХ МЕРЕЖ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися створювати та застосовувати прості нейронні мережі**.**

Завдання 2.1 Створити простий нейрон

```python
import numpy as np
def sigmoid(x):
    # Наша функція активації: f(x) = 1 / (1 + e^(-x))
    return 1 / (1 + np.exp(-x))
class Neuron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias
    def feedforward(self, inputs):
    # Вхідні дані про вагу, додавання зміщення
    # і подальше використання функції активації
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)
weights = np.array([0, 1])  # w1 = 0, w2 = 1
bias = 4  # b = 4
n = Neuron(weights, bias)
x = np.array([2, 3])  # x1 = 2, x2 = 3
print(n.feedforward(x))
```

Рис 5.1 Код файлу LR_5_task1.py

```
0.9990889488055994

Process finished with exit code 0
```

Рис 5.2 Результат файлу LR_5_task1.py

Завдання 2.2. Створити просту нейронну мережу для передбачення статі людини

```python
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
class Neuron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias
    def feedforward(self, inputs):
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)
weights = np.array([0, 1])  # w1 = 0, w2 = 1
bias = 4  # b = 4
n = Neuron(weights, bias)
x = np.array([2, 3])  # x1 = 2, x2 = 3
class NadvornyiNeuralNetwork:
    def __init__(self):
        weights = np.array([0, 1])
        bias = 0
        # Класс Neuron із попереднього завдання
        self.h1 = Neuron(weights, bias)
        self.h2 = Neuron(weights, bias)
        self.o1 = Neuron(weights, bias)
    def feedforward(self, x):
        out_h1 = self.h1.feedforward(x)
        out_h2 = self.h2.feedforward(x)
        # Входи для o1 є виходами h1 и h2
        out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))
        return out_o1
network = NadvornyiNeuralNetwork()
x = np.array([2, 3])
```

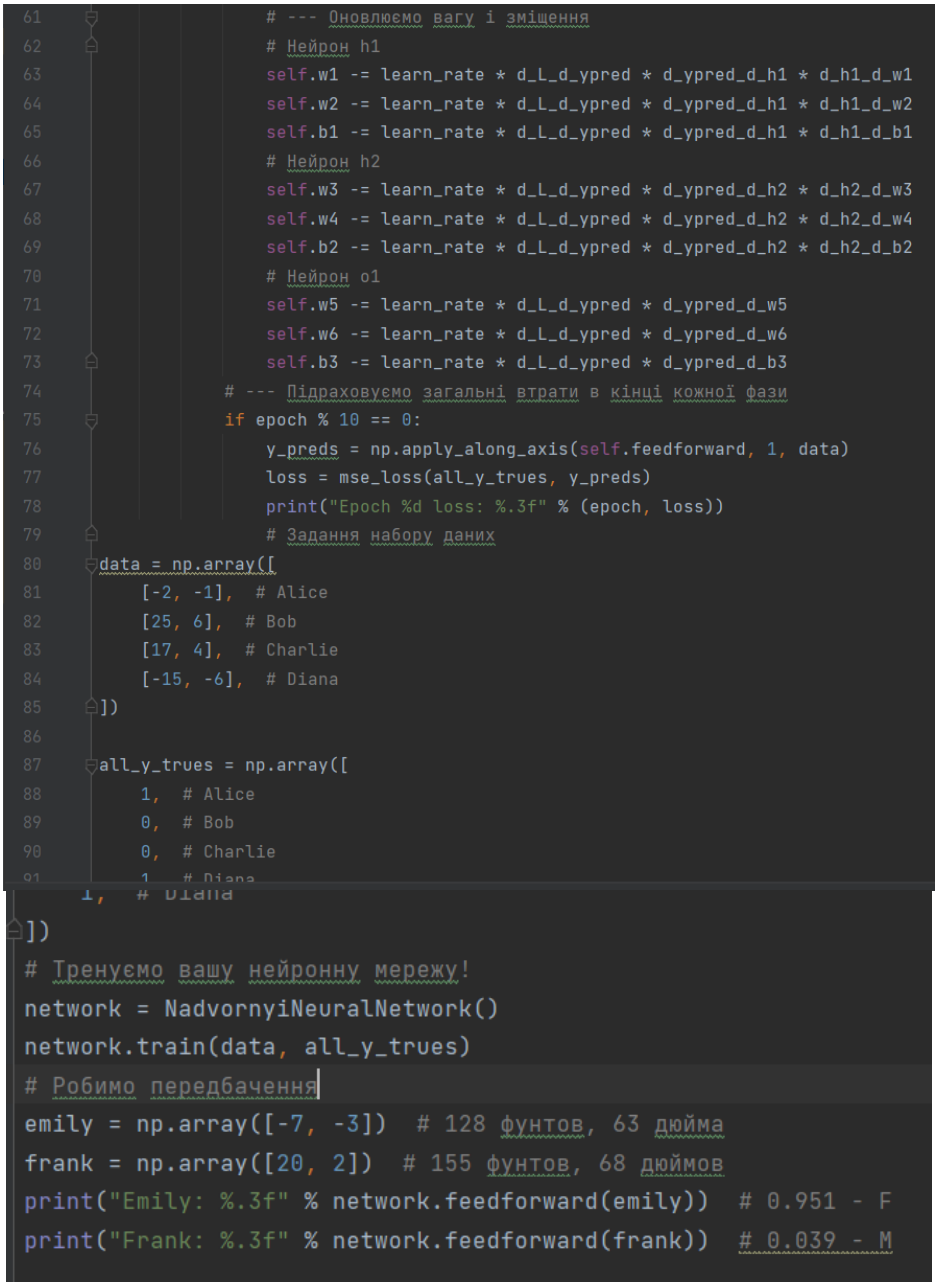
Рис 5.3 Код файлу LR_5_task_2_1.py

```
0.7216325609518421
```

Рис 5.4 Результат файлу LR_5_task_2_1.py

```python
import numpy as np
def sigmoid(x):
    # Функція активації sigmoid:: f(x) = 1 / (1 + e^(-x))
    return 1 / (1 + np.exp(-x))
def deriv_sigmoid(x):
    # Похідна від sigmoid: f'(x) = f(x) * (1 - f(x))
    fx = sigmoid(x)
    return fx * (1 - fx)
def mse_loss(y_true, y_pred):
    # y_true и y_pred є масивами numpy з одинаковою довжиною
    return ((y_true - y_pred) ** 2).mean()
class NadvornyiNeuralNetwork:
    def __init__(self):
        #Вага
        self.w1 = np.random.normal()
        self.w2 = np.random.normal()
        self.w3 = np.random.normal()
        self.w4 = np.random.normal()
        self.w5 = np.random.normal()
        self.w6 = np.random.normal()
        #Зміщення
        self.b1 = np.random.normal()
        self.b2 = np.random.normal()
        self.b3 = np.random.normal()
    def feedforward(self, x):
        # x є масивом numpy з двома елементами
        h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
        h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
        o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
        return o1
    def train(self, data, all_y_trues):
```

```python
    def train(self, data, all_y_trues):
        learn_rate = 0.1
        epochs = 1000
        for epoch in range(epochs):
            for x, y_true in zip(data, all_y_trues):
                # --- Виконуємо зворотній зв'язок (ці значання нам потрібні в подальшому
                sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
                h1 = sigmoid(sum_h1)
                sum_h2 = self.w3 * x[0] + self.w4 * x[1] + self.b2
                h2 = sigmoid(sum_h2)
                sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
                o1 = sigmoid(sum_o1)
                y_pred = o1
                # --- Підрахунок часткових похідних
                # --- Найменування: d_L_d_w1 означає "частково L / частково w1"
                d_L_d_ypred = -2 * (y_true - y_pred)
                # Нейрон o1
                d_ypred_d_w5 = h1 * deriv_sigmoid(sum_o1)
                d_ypred_d_w6 = h2 * deriv_sigmoid(sum_o1)
                d_ypred_d_b3 = deriv_sigmoid(sum_o1)
                d_ypred_d_h1 = self.w5 * deriv_sigmoid(sum_o1)
                d_ypred_d_h2 = self.w6 * deriv_sigmoid(sum_o1)
                # Нейрон h1
                d_h1_d_w1 = x[0] * deriv_sigmoid(sum_h1)
                d_h1_d_w2 = x[1] * deriv_sigmoid(sum_h1)
                d_h1_d_b1 = deriv_sigmoid(sum_h1)
                # Нейрон h2
                d_h2_d_w3 = x[0] * deriv_sigmoid(sum_h2)
                d_h2_d_w4 = x[1] * deriv_sigmoid(sum_h2)
                d_h2_d_b2 = deriv_sigmoid(sum_h2)
                # --- Оновлюємо вагу і зміщення
```



```
                # --- Оновлюємо вагу і зміщення
                # Нейрон h1
                self.w1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w1
                self.w2 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w2
                self.b1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_b1
                # Нейрон h2
                self.w3 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w3
                self.w4 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w4
                self.b2 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_b2
                # Нейрон o1
                self.w5 -= learn_rate * d_L_d_ypred * d_ypred_d_w5
                self.w6 -= learn_rate * d_L_d_ypred * d_ypred_d_w6
                self.b3 -= learn_rate * d_L_d_ypred * d_ypred_d_b3
            # --- Підраховуємо загальні втрати в кінці кожної фази
            if epoch % 10 == 0:
                y_preds = np.apply_along_axis(self.feedforward, 1, data)
                loss = mse_loss(all_y_trues, y_preds)
                print("Epoch %d loss: %.3f" % (epoch, loss))
# Задання набору даних
data = np.array([
    [-2, -1],  # Alice
    [25, 6],  # Bob
    [17, 4],  # Charlie
    [-15, -6],  # Diana
])

all_y_trues = np.array([
    1,  # Alice
    0,  # Bob
    0,  # Charlie
    1,  # Diana
])
# Тренуємо вашу нейронну мережу!
network = NadvornyiNeuralNetwork()
network.train(data, all_y_trues)
# Робимо передбачення
emily = np.array([-7, -3])  # 128 фунтов, 63 дюйма
frank = np.array([20, 2])  # 155 фунтов, 68 дюймов
print("Emily: %.3f" % network.feedforward(emily))  # 0.951 - F
print("Frank: %.3f" % network.feedforward(frank))  # 0.039 - M
```

Рис 5.5 Код файлу LR_5_task_2_2.py

```
Epoch 0 loss: 0.421
Epoch 10 loss: 0.070
Epoch 20 loss: 0.054
Epoch 30 loss: 0.044
Epoch 40 loss: 0.036
Epoch 50 loss: 0.031
Epoch 60 loss: 0.026
Epoch 70 loss: 0.023
Epoch 80 loss: 0.020
Epoch 90 loss: 0.018
Epoch 100 loss: 0.017
```

```
Epoch 110 loss: 0.015
Epoch 120 loss: 0.014
Epoch 130 loss: 0.013
Epoch 140 loss: 0.012
Epoch 150 loss: 0.011
Epoch 160 loss: 0.010
Epoch 170 loss: 0.010
Epoch 180 loss: 0.009
Epoch 190 loss: 0.009
Epoch 200 loss: 0.008
Epoch 210 loss: 0.008
Epoch 220 loss: 0.007
Epoch 230 loss: 0.007
Epoch 240 loss: 0.007
Epoch 250 loss: 0.006
Epoch 260 loss: 0.006
Epoch 270 loss: 0.006
Epoch 280 loss: 0.006
Epoch 290 loss: 0.005
Epoch 300 loss: 0.005
Epoch 310 loss: 0.005
Epoch 320 loss: 0.005
Epoch 330 loss: 0.005
Epoch 340 loss: 0.005
Epoch 350 loss: 0.004
Epoch 360 loss: 0.004
Epoch 370 loss: 0.004
Epoch 380 loss: 0.004
Epoch 390 loss: 0.004
Epoch 400 loss: 0.004
Epoch 410 loss: 0.004
Epoch 420 loss: 0.004
Epoch 430 loss: 0.004
Epoch 440 loss: 0.004
Epoch 450 loss: 0.003
Epoch 460 loss: 0.003
Epoch 470 loss: 0.003
Epoch 480 loss: 0.003
Epoch 490 loss: 0.003
Epoch 500 loss: 0.003
Epoch 510 loss: 0.003
Epoch 520 loss: 0.003
Epoch 530 loss: 0.003
Epoch 540 loss: 0.003
Epoch 550 loss: 0.003
Epoch 560 loss: 0.003
Epoch 570 loss: 0.003
Epoch 580 loss: 0.003
Epoch 590 loss: 0.003
Epoch 600 loss: 0.003
Epoch 610 loss: 0.002
Epoch 620 loss: 0.002
Epoch 630 loss: 0.002
Epoch 640 loss: 0.002
```

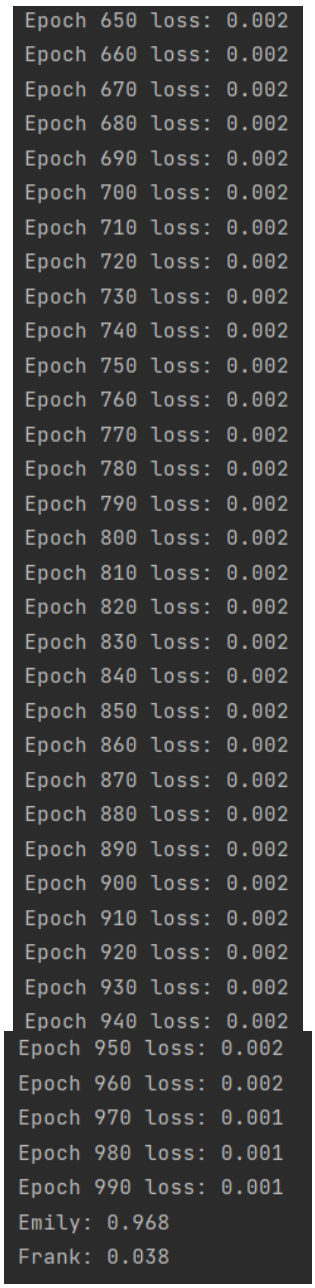


Рис 5.6 Результат файлу LR_5_task_2_2.py

Висновок: Функція активації, або передавальна функція штучного нейрона — залежність вихідного сигналу штучного нейрона від вхідного. Більшість видів нейронних мереж для функції активації використовують сигмоїди.

Можливості нейроннних мереж прямого поширення полягають в тому, що сигнали поширюються в одному напрямку, починаючи від вхідного шару нейронів, через приховані шари до вихідного шару і на вихідних нейронах отримується результат опрацювання сигналу. В мережах такого виду немає зворотніх зв’язків.

Нейронні мережі прямого поширення знаходять своє застосування в задачах ком-п'ютерного бачення та розпізнаванні мовлення, де класифікація цільових класів ускладнюється. Такі типи нейронних мереж добре справляються із зашумленими даними.

Завдання 2.3. Класифікатор на основі перцептрону з використанням бібліотеки NeuroLab

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
# Завантаження вхідних даних
text = np.loadtxt('data_perceptron.txt')
# Поділ точок даних та міток
data = text[:, :2]
labels = text[:, 2].reshape((text.shape[0], 1))
# Побудова графіка вхідних даних
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
# Визначення максимального та мінімального значень для кожного виміру
dim1_min, dim1_max, dim2_min, dim2_max = 0, 1, 0, 1
# Кількість нейронів у вихідному шарі
num_output = labels.shape[1]
# Визначення перцептрону з двома вхідними нейронами (оскільки
# Вхідні дані - двовимірні)
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
perceptron = nl.net.newp([dim1, dim2], num_output)
# Тренування перцептрону з використанням наших даних
error_progress = perceptron.train(data, labels, epochs = 100, show = 20, lr = 0.03)
# Побудова графіка процесу навчання
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
plt.grid()
plt.show()
```

Рис 5.7 Код файлу LR_5_task_3.py

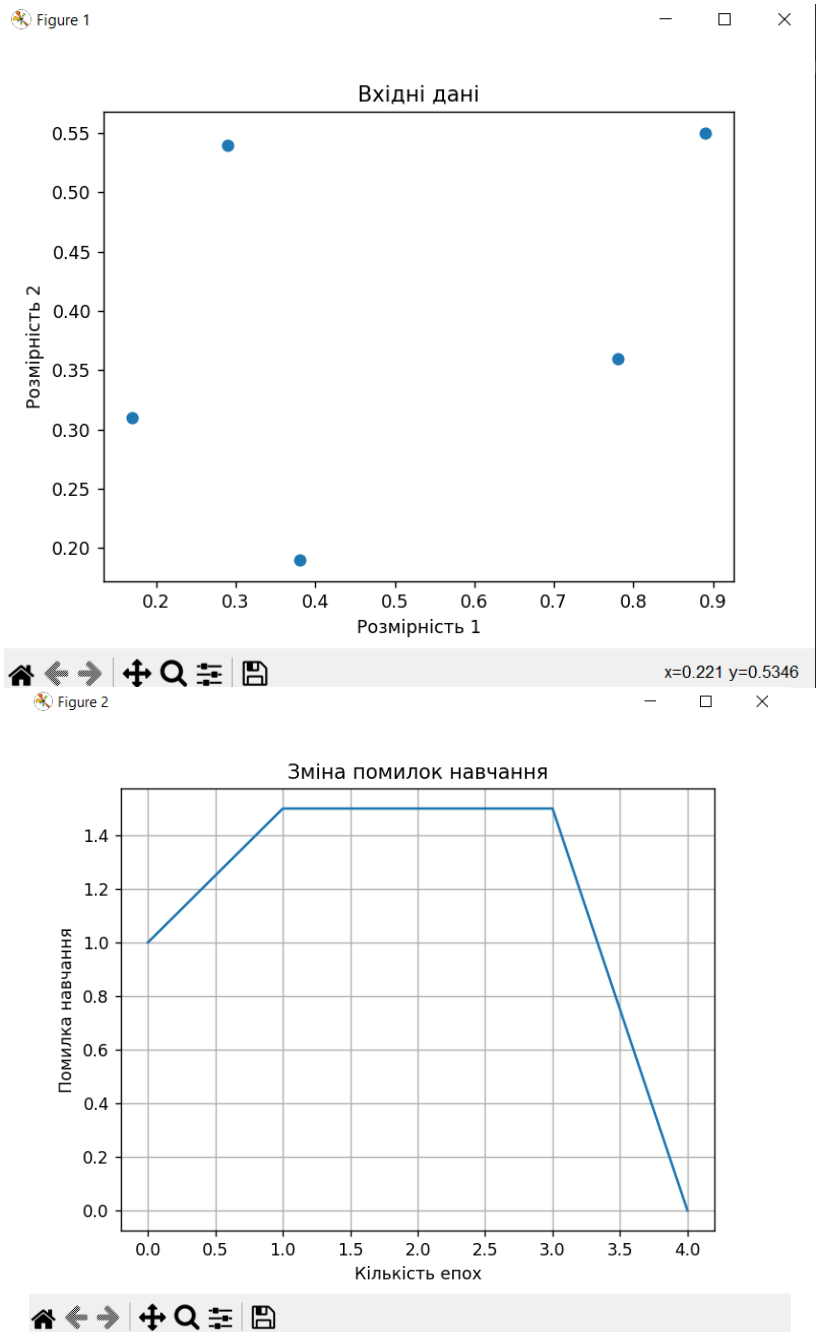
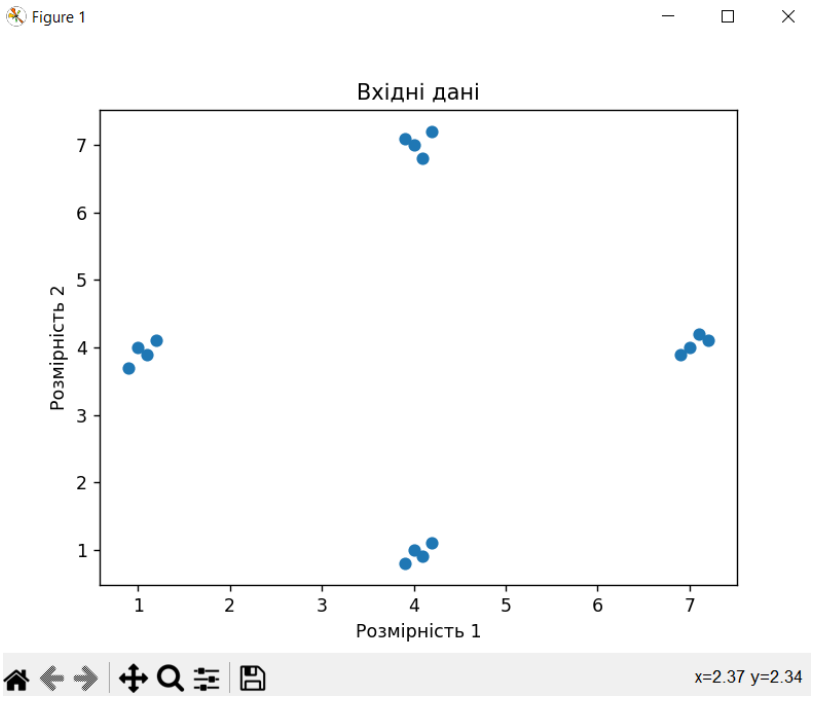


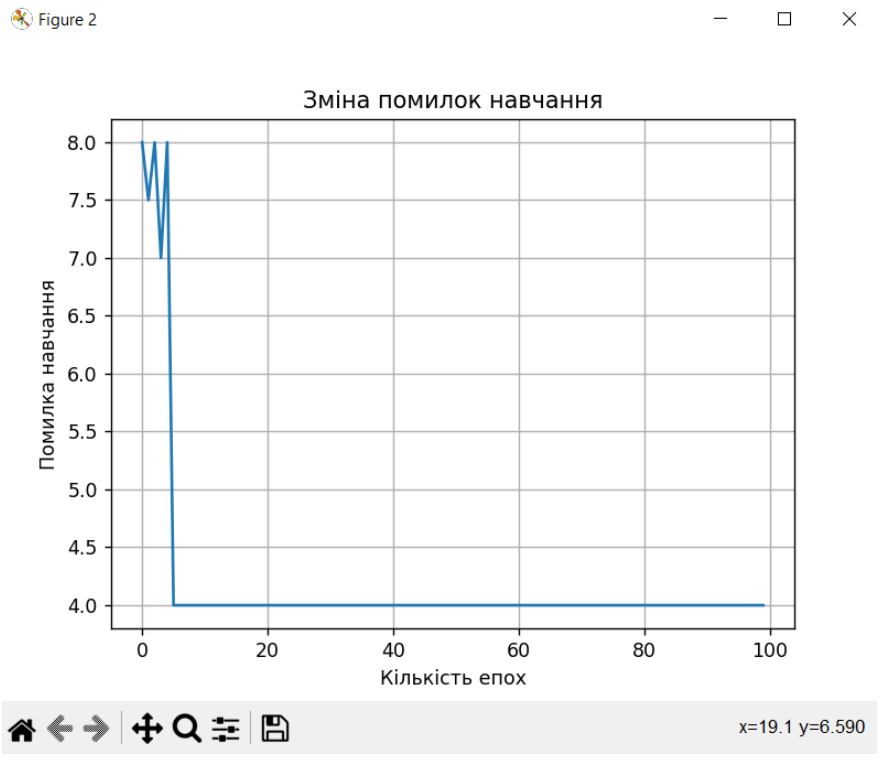
Рис 5.8 Результат файлу LR_5_task_3.py

Завдання 2.4. Побудова одношарової нейронної мережі

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
text = np.loadtxt('data_simple_nn.txt')
data = text[:, 0:2]
labels = text[:, 2:]
plt.figure()
plt.scatter(data[:, 0], data[:, 1])
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
dim1_min, dim1_max = data[:, 0].min(), data[:, 0].max()
dim2_min, dim2_max = data[:, 1].min(), data[:, 1].max()
num_output = labels.shape[1]
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
nn = nl.net.newp([dim1, dim2], num_output)
error_progress = nn.train(data, labels, epochs = 100, show = 20, lr = 0.03)
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
plt.grid()
plt.show()
print('\nTest results:')
data_test = [[0.4, 4.3], [4.4, 0.6], [4.7, 8.1]]
for item in data_test:
    print(item, '-->', nn.sim([item])[0])
```

Рис 5.9 Код файлу LR_5_task_4.py



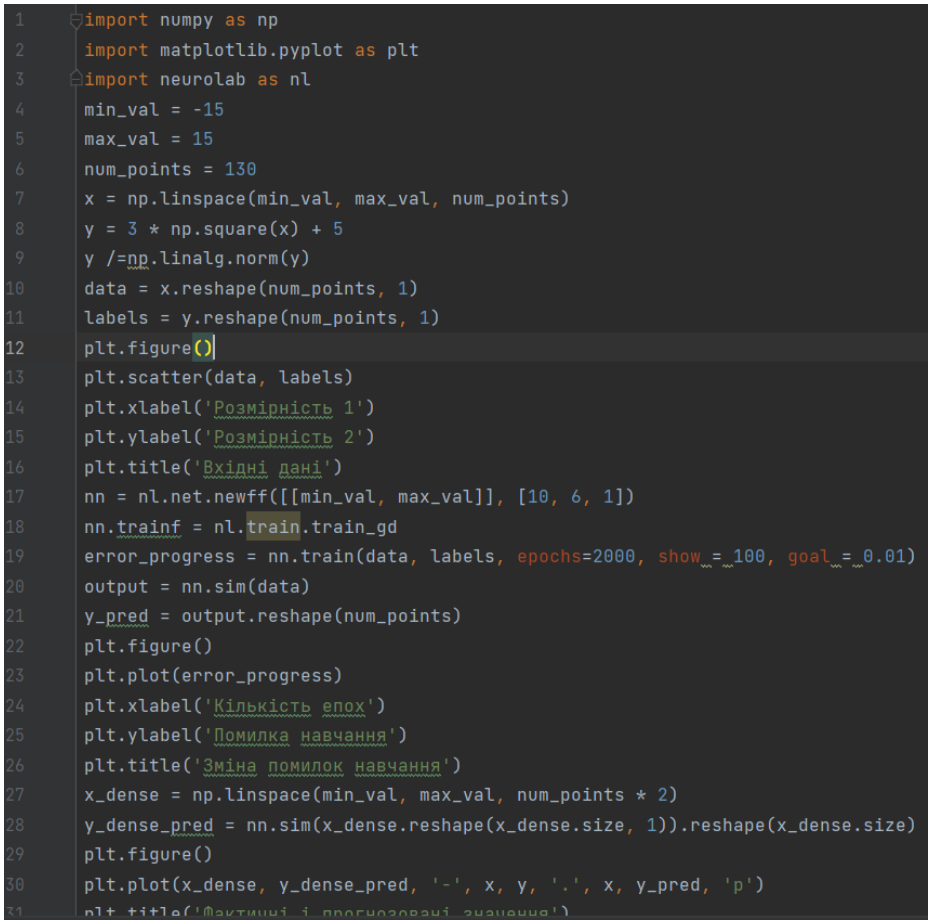



```
Epoch: 20; Error: 4.0;
Epoch: 40; Error: 4.0;
Epoch: 60; Error: 4.0;
Epoch: 80; Error: 4.0;
Epoch: 100; Error: 4.0;
The maximum number of train epochs is reached

Test results:
[0.4, 4.3] --> [0. 0.]
[4.4, 0.6] --> [1. 0.]
[4.7, 8.1] --> [1. 1.]

Process finished with exit code 0
```

Рис 5.10 Результат файлу LR_5_task_4.py

Висновок: На рис. 20 зображено процес навчання мережі. На 20 епосі відбулось 4 помилки, аналогічно на 40, 60, 80 та 100. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування. Ми вирішили визначити вибіркові тестові точки даних та запустили для них нейронну мережу. І це його результат.

Завдання 2.5. Побудова багатошарової нейронної мережі



```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points)
y = 3 * np.square(x) + 5
y /=np.linalg.norm(y)
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
nn = nl.net.newff([[min_val, max_val]], [10, 6, 1])
nn.trainf = nl.train.train_gd
error_progress = nn.train(data, labels, epochs=2000, show = 100, goal = 0.01)
output = nn.sim(data)
y_pred = output.reshape(num_points)
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
x_dense = np.linspace(min_val, max_val, num_points * 2)
y_dense_pred = nn.sim(x_dense.reshape(x_dense.size, 1)).reshape(x_dense.size)
plt.figure()
plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
plt.title('Фактичні і прогнозовані значення')
```

Рис 5.11 Код файлу LR_5_task_5.py

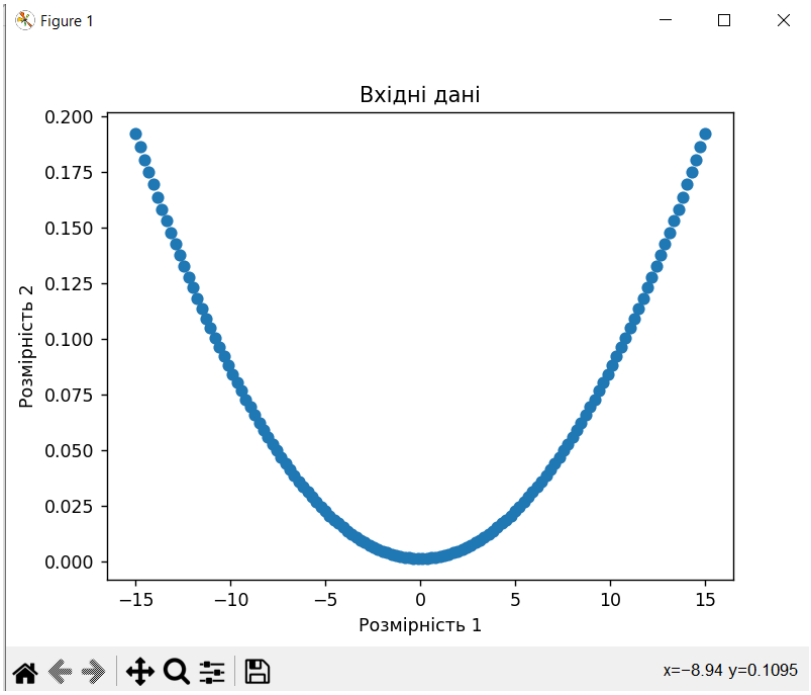


Figure 1
Вхідні дані
Розмірність 2
Розмірність 1
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−15
−10
−5
0
5
10
15
x=−8.94 y=0.1095

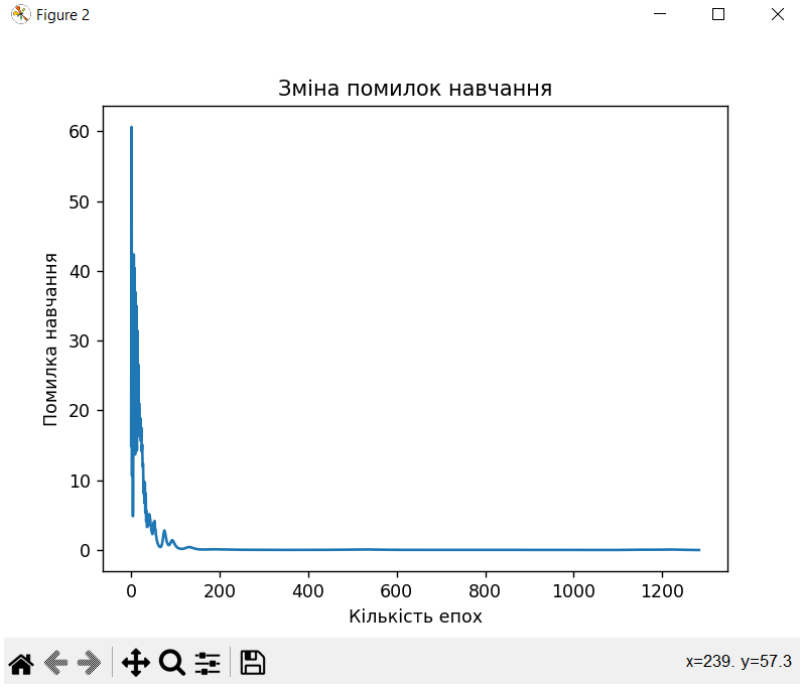
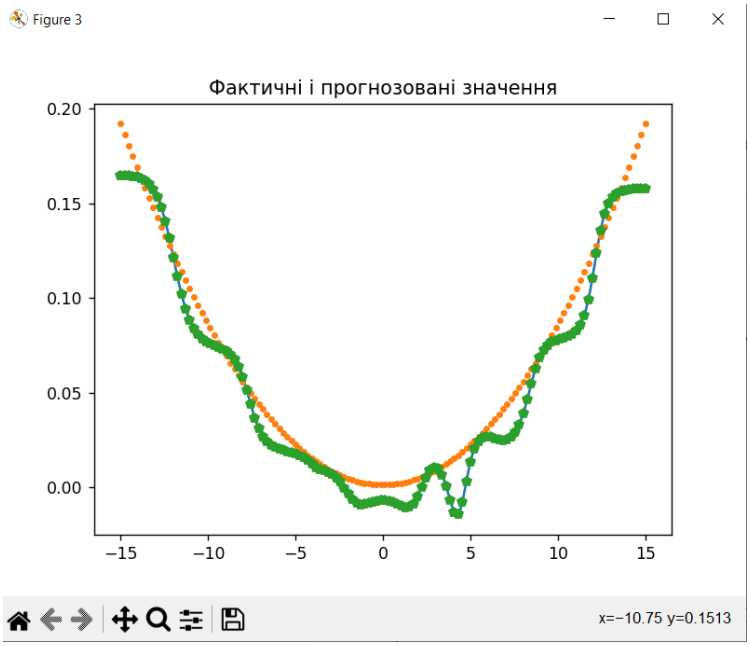


Figure 2
Зміна помилок навчання
Помилка навчання
Кількість епох
60
50
40
30
20
10
0
0
200
400
600
800
1000
1200
x=239. y=57.3




```
Epoch: 100; Error: 0.8262482672508197;
Epoch: 200; Error: 0.10774723057552665;
Epoch: 300; Error: 0.032067132677348884;
Epoch: 400; Error: 0.027557832453375293;
Epoch: 500; Error: 0.09009621058390158;
Epoch: 600; Error: 0.03814733492157424;
Epoch: 700; Error: 0.02742544367785592;
Epoch: 800; Error: 0.04180327766927411;
Epoch: 900; Error: 0.02670507858213031;
Epoch: 1000; Error: 0.025459895724339 94;
Epoch: 1100; Error: 0.019261256834020617;
Epoch: 1200; Error: 0.062013469962181;
The goal of learning is reached
```

Рис 5.12 Результат файлу LR_5_task_5.py

Висновок: На рис. 12 зображено процес навчання мережі. Відносно кожної епосі відбувались помилки. На 100 0.83 помилки. На 1200 0.06. Потім вивелось повідомлення, що ми досягли цілі навчання.

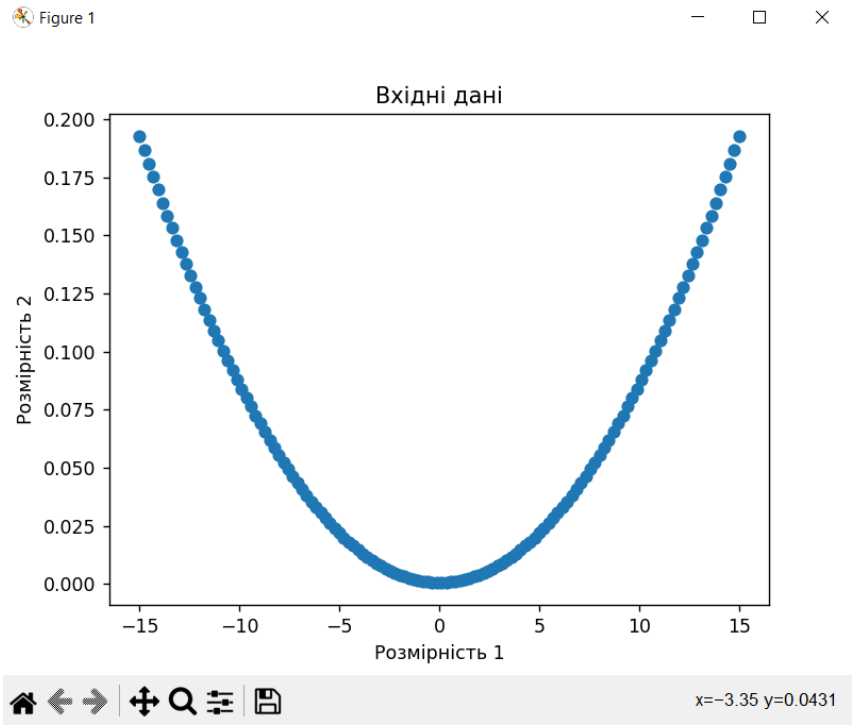
Завдання 2.6. Побудова багатошарової нейронної мережі для свого варіанту

Варіант 12	$y = 5x^2+3$	Варіант 27	$y = 4x^2+3x+2$

12	3	3-4-1

```python
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
min_val = -15
max_val = 15
num_points = 130
x = np.linspace(min_val, max_val, num_points)
y = 5 * x * x + 3
y /=np.linalg.norm(y)
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Розмірність 1')
plt.ylabel('Розмірність 2')
plt.title('Вхідні дані')
nn = nl.net.newff([[min_val, max_val]], [3, 4, 1])
nn.trainf = nl.train.train_gd
error_progress = nn.train(data, labels, epochs=2000, show = 100, goal = 0.01)
output = nn.sim(data)
y_pred = output.reshape(num_points)
plt.figure()
plt.plot(error_progress)
plt.xlabel('Кількість епох')
plt.ylabel('Помилка навчання')
plt.title('Зміна помилок навчання')
x_dense = np.linspace(min_val, max_val, num_points * 2)
y_dense_pred = nn.sim(x_dense.reshape(x_dense.size, 1)).reshape(x_dense.size)
plt.figure()
plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
plt.title('Фактичні і прогнозовані значення')
```

Рис 5.13 Код файлу LR_5_task_6.py



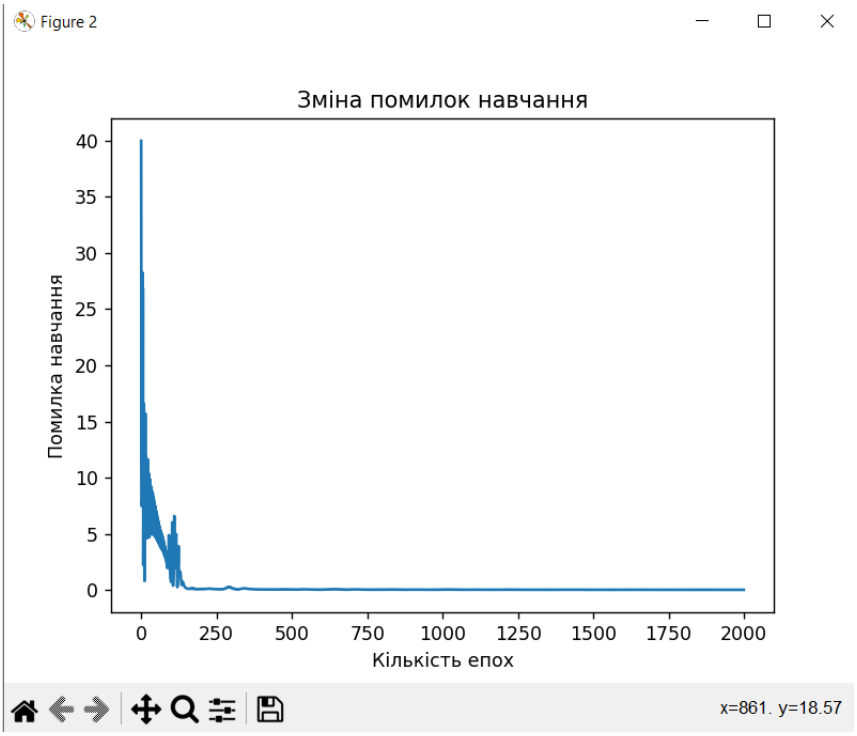


Figure 2
Зміна помилок навчання
Помилка навчання
Кількість епох
0
5
10
15
20
25
30
35
40
0
250
500
750
1000
1250
1500
1750
2000
x=861. y=18.57

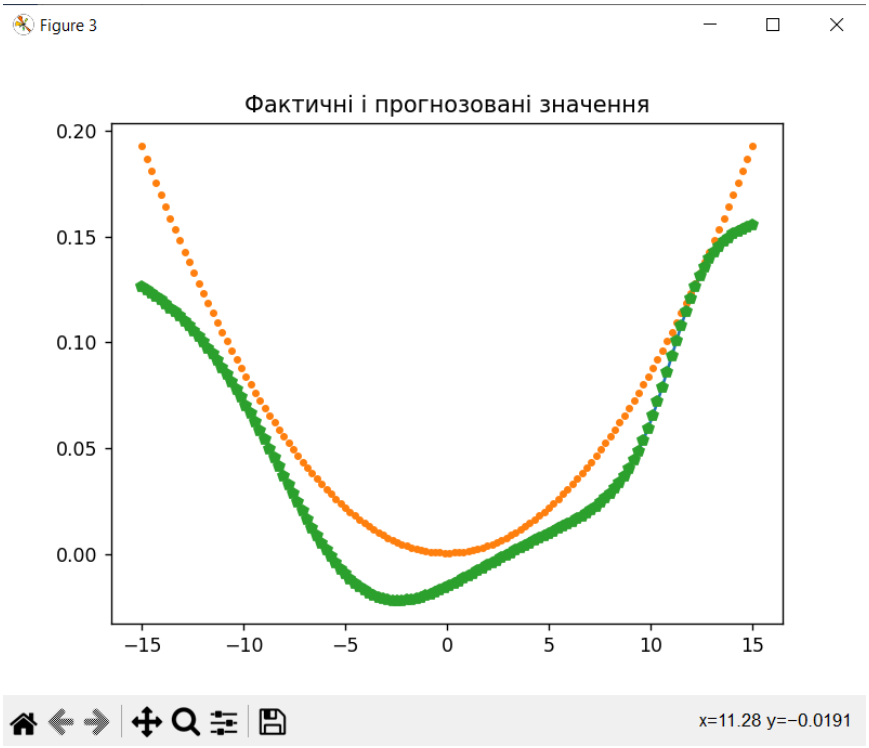


Figure 3
Фактичні і прогнозовані значення
0.20
0.15
0.10
0.05
0.00
−15
−10
−5
0
5
10
15
x=11.28 y=−0.0191

```
Epoch: 100; Error: 0.7518501572562898;
Epoch: 200; Error: 0.10977857305233404;
Epoch: 300; Error: 0.21039031091940777;
Epoch: 400; Error: 0.06963863652324795;
Epoch: 500; Error: 0.06451102571080759;
Epoch: 600; Error: 0.057330599299357285;
Epoch: 700; Error: 0.07105299397950451;
Epoch: 800; Error: 0.05423377978619075;
Epoch: 900; Error: 0.053822978903777366;
Epoch: 1000; Error: 0.05480456381351653;
Epoch: 1100; Error: 0.05200600106100691;
Epoch: 1200; Error: 0.04957354433720307;
Epoch: 1300; Error: 0.04487585454246928;
Epoch: 1400; Error: 0.04541434157214551;
Epoch: 1500; Error: 0.041842645188862;
Epoch: 1600; Error: 0.041998422454544916;
Epoch: 1700; Error: 0.040894779318702376;
Epoch: 1800; Error: 0.03983116109383445;
Epoch: 1900; Error: 0.04023918712131063;
Epoch: 2000; Error: 0.038563099604737056;
The maximum number of train epochs is reached
```

Рис 5.14 Результат файлу LR_5_task_6.py

На рис. 14 зображено процес навчання мережі. На 100 епосі відбулось 0.75 помилки, на 300 епосі відбулось 0.21 помилки, і так далі, на 2000 епосі відбулось 0.03 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

Завдання 2.7. Побудова нейронної мережі на основі карти Кохонена, що самоорганізується

```python
import numpy as np
import neurolab as nl
import numpy.random as rand
skv = 0.05
centr = np.array([[0.2, 0.2], [0.4, 0.4], [0.7, 0.3], [0.2, 0.5]])
rand_norm = skv * rand.randn(100, 4, 2)
inp = np.array([centr + r for r in rand_norm])
inp.shape = (100 * 4, 2)
rand.shuffle(inp)
# Create net with 2 inputs and 4 neurons
net = nl.net.newc([[0.0, 1.0],[0.0, 1.0]], 4)
# train with rule: Conscience Winner Take All algoritm (CWTA)
error = net.train(inp, epochs=200,  show=100)
# Plot results:
import pylab as pl
pl.title('Classification Problem')
pl.subplot(211)
pl.plot(error)
pl.xlabel('Epoch number')
pl.ylabel('error (default MAE)')
w = net.layers[0].np['w']
pl.subplot(212)
pl.plot(inp[:,0], inp[:,1], '.',\
        centr[:,0], centr[:, 1],, 'yv',\
        w[:,0], w[:,1], 'p')
pl.legend(['train samples', 'real centers', 'train centers'])
pl.show()
```

Рис 5.15 Код файлу LR_5_task_7.py

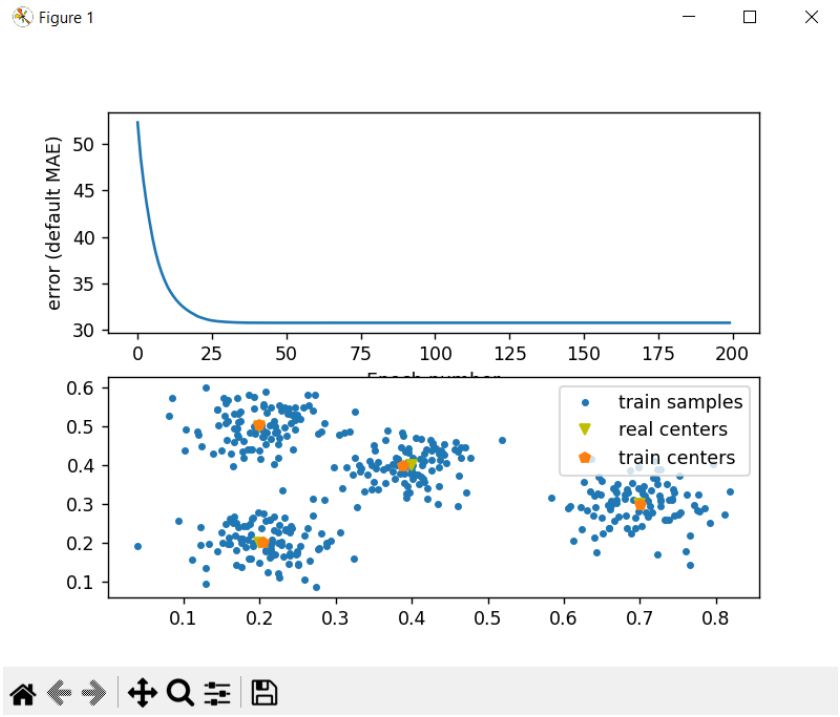


Рис 5.16 Результат файлу LR_5_task_7.py

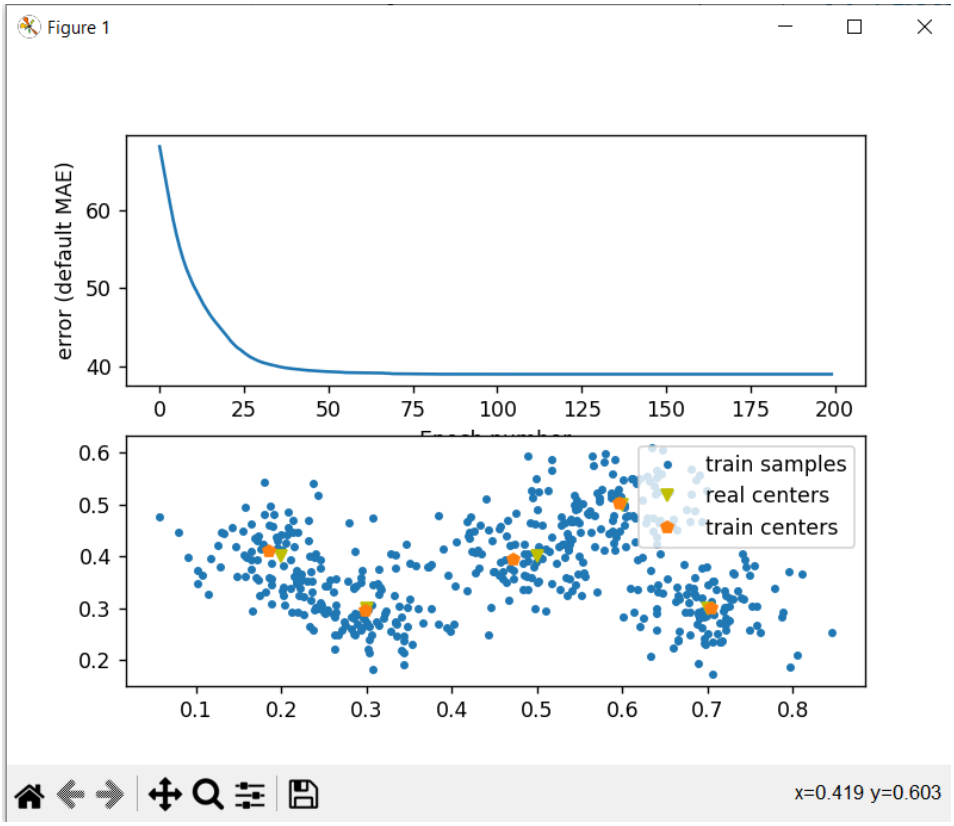
Помилка MAE - Средня абсолютна помилка (Mean Absolute Error). Середньою абсолютною похибкою називають середнє арифметичне з абсолютних похибок усіх вимірювань.

Завдання 2.8. Дослідження нейронної мережі на основі карти Кохонена, що самоорганізується

Варіант 12	[0.3, 0.3], [0.5, 0.4], [0.7, 0.3], [0.2, 0.4], [0.6, 0.5]	0,05

```python
import numpy as np
import neurolab as nl
import numpy.random as rand
skv = 0.05
centr = np.array([[0.3, 0.3], [0.5, 0.4], [0.7, 0.3], [0.2, 0.4], [0.6, 0.5]])
rand_norm = skv * rand.randn(100, 5, 2)
inp = np.array([centr + r for r in rand_norm])
inp.shape = (100 * 5, 2)
rand.shuffle(inp)
# Create net with 2 inputs and 5 neurons
net = nl.net.newc([[0.0, 1.0],[0.0, 1.0]], 5)
# train with rule: Conscience Winner Take All algoritm (CWTA)
error = net.train(inp, epochs=200,  show=20)
# Plot results:
import pylab as pl
pl.title('Classification Problem')
pl.subplot(211)
pl.plot(error)
pl.xlabel('Epoch number')
pl.ylabel('error (default MAE)')
w = net.layers[0].np['w']
pl.subplot(212)
pl.plot(inp[:,0], inp[:,1], '.',\
        centr[:,0], centr[:, 1], 'yv',\
        w[:,0], w[:,1], 'p')
pl.legend(['train samples', 'real centers', 'train centers'])
pl.show()
```

Рис 5.17 Код файлу LR_5_task_8.py

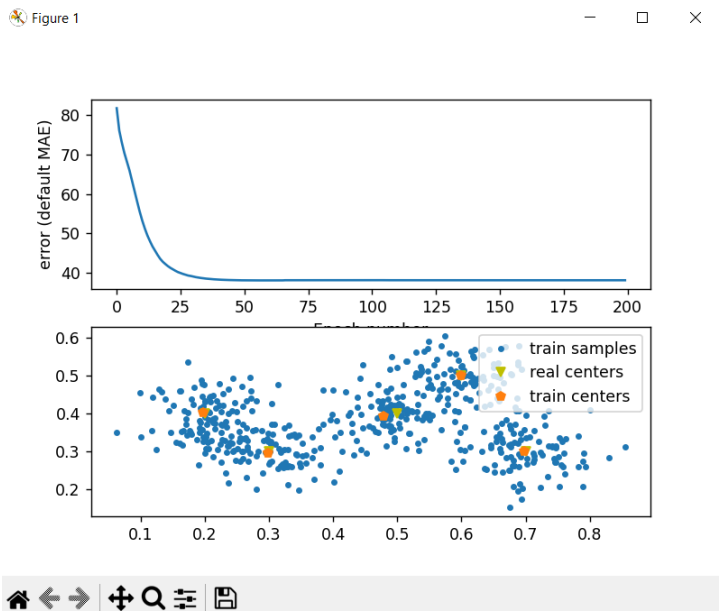


```
Epoch: 20; Error: 44.3627304620524;
Epoch: 40; Error: 39.71935701904583;
Epoch: 60; Error: 39.18708719624229;
Epoch: 80; Error: 39.02269788242847;
Epoch: 100; Error: 39.004689906812914;
Epoch: 120; Error: 39.00547783304149;
Epoch: 140; Error: 39.00551839434286;
Epoch: 160; Error: 39.00547531607146;
Epoch: 180; Error: 39.005459308978516;
Epoch: 200; Error: 39.0054550509531;
The maximum number of train epochs is reached
```

Рис 5.18 Результат файлу LR_5_task_8.py

На рис. 18 зображено процес навчання мережі. На 20 епосі відбулось 44.36 помилки, помилки і так далі, на 200 епосі відбулось 39.01 помилки,. Потім вивелось повідомлення, що ми досягли максимальної кількості епох для тренування.

```
# Create net with 2 inputs and 5 neurons
net = nl.net.newc([[0.0, 1.0],[0.0, 1.0]], 5)
```



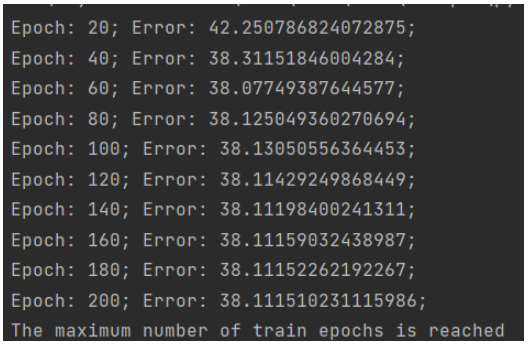


Рис 5.19 Результат файлу LR_5_task_8.py

Якщо порівнювати нейронну мережу Кохонена з 4 нейронами та 5 нейронами, можна зробити такі висновки. При 4 нейронах Помилка MAE повільніше зменшується, ніж з 5 нейронами, також з 5 нейронами ця помилка нижча. З 5 нейронами обоє центрів збігаються майже в одні точці. Число нейронів в шарі Кохонена має відповідати числу класів вхідних сигналів. Тобто в нашому випадку нам давалось 5 вхідних сигналів, значить у нас має бути 5 нейронів, а не 4.Отже, невірний вибір кількості нейронів числу кластерів впливає на величину помилки ускладнюючи навчання мережі і швидкості, тому на рис. 18 набагато гірші результати, ніж на рис. 19.

Висновок: на лабораторній роботі я навчився використовувати спеціалізовані бібліотеки та мову програмування Python, навчився створювати та застосовувати прості нейронні мережі**.**

Посилання на гітхаб: https://github.com/Max1648/Artificial-Intelligence2